Threat Detection and Containment

To simulate **Threat Detection and Containment** for your Cloud Immune System (CIS) on AWS, here's a **step-by-step guide** for the scenario you described. This setup uses AWS Free Tier–eligible services (as much as possible), and works with Jupyter Notebooks for parts of the data processing and SageMaker ML integration.

✅ Prerequisites:

- AWS Free Tier account (already confirmed)
- IAM user with AdminAccess
- Enabled AWS Regions (e.g., us-east-1)
- Jupyter Notebook (inside SageMaker Studio or local)

🔧

Step-by-Step Implementation

---

1. EC2 Instance Setup (Target for Compromise Simulation)

- Go to **EC2 Dashboard**
- Launch a **t2.micro** instance with Amazon Linux or Ubuntu
- Create and assign a Security Group with SSH and limited outbound access
- Tag the instance: Name = CompromisedInstance

---

2. Enable VPC Flow Logs

- Go to **VPC > Your VPC > Flow Logs > Create Flow Log**
- Select your VPC
- Destination: **CloudWatch Logs**
- IAM Role: Create one with vpc-flow-logs-role permissions
- Filter: All traffic
- Purpose: This captures outbound traffic patterns

---

3. Simulate Malicious Outbound Traffic

- SSH into your EC2 instance
- Use curl or wget to simulate data exfiltration:

Bash

```
for i in {1..20}; do curl http://example.com --output /dev/null; sleep 5; done
```

This will produce repetitive outbound connections to an external domain.

---

4. Create a SageMaker Anomaly Detection Model

- Open SageMaker Studio or Notebooks
- Train an **Autoencoder** or use built-in anomaly detection algorithm using synthetic data

Python

```
# Sample flow log features: source IP, destination IP, bytes sent, connection duration
import pandas as pd
from sklearn.ensemble import IsolationForest

df = pd.read_csv('vpc_flow_logs_sample.csv')  # Simulated or real logs
model = IsolationForest(contamination=0.1).fit(df)
predictions = model.predict(df)
```

- Deploy the model using **SageMaker Endpoint**

5. Automate Containment with Lambda and EventBridge

➢ **Create EventBridge Rule:**

- Go to **EventBridge > Rules > Create rule**
- Source: Custom (or set to CloudWatch Logs Metric Filter)
- Pattern: Monitor abnormal outbound traffic event or SageMaker model output
- Target: Lambda function

➢ **Create Lambda Function:**

```
import boto3

def lambda_handler(event, context):
   ec2 = boto3.client('ec2')
   instance_id = 'i-0abcdef1234567890'  # Replace with real ID

   # Modify security group to block all traffic
   ec2.modify_instance_attribute(InstanceId=instance_id, Groups=['sg-xxxxxxx'])  # Empty SG
   return {"status": "EC2 Isolated"}
```

- Assign a basic execution role with EC2 permissions

---

6. Set up Amazon SNS for Notifications

- Go to **SNS > Topics > Create topic**
- Choose Standard
- Name: CIS-Alert
- Create Subscription (Email)

- In Lambda, add:

```
Python
sns = boto3.client('sns')
sns.publish(TopicArn='arn:aws:sns:region:acct-id:CIS-Alert',
            Subject='Alert: EC2 Isolated',
            Message='Suspicious traffic detected. Instance has been isolated.')
```

📊 Optional Dashboard:

Use **Amazon QuickSight** or open-source tools like **Grafana + CloudWatch** for visualizing metrics like:

- Alert frequency
- Isolation latency
- Detection success rate

✅ Summary of What You Achieve:

Component	Purpose
EC2 + VPC Logs	Simulated attack target
SageMaker	Anomaly detection
EventBridge + Lambda	Automated response
SNS	Real-time notification

S3 + Glue	Retraining data pipeline